PLC Logic-Based Cybersecurity Risks Identification for ICS

Mike Da Silva, Maxime Puys, Pierre-Henri Thevenon, Stéphane Mocanu

PLC Logic-Based Cybersecurity Risks Identification for ICS

Mike Da Silva
CEA-Leti, Université Grenoble Alpes
Grenoble, France
mike.dasilva@cea.fr

Maxime Puys
CEA-Leti, Université Grenoble Alpes
Grenoble, France
maxime.puys@cea.fr

Pierre-Henri Thevenon
CEA-Leti, Université Grenoble Alpes
Grenoble, France
pierre-henri.thevenon@cea.fr

Stéphane Mocanu
Laboratoire d'Informatique de Grenoble
Univ. Grenoble Alpes, CNRS, Inria, Grenoble-INP
Grenoble, France
stephane.mocanu@inria.fr

ABSTRACT

In recent years, Informational Technologies (IT) was massively deployed into Industrial Control Systems (ICS) mainly for its economic benefits. However, this new paradigm, converging IT and Operational Technologies (OT), brings new challenges that companies need to face. Historically, ICS had to cope with safety requirements which ensure the protection of people, environment, and assets. Now, ICS must deal with additional threats, coming from cyberattacks, in order to maintain safety. For that purpose, it becomes essential to develop new cybersecurity technologies and methodologies that allow to assess the safety of ICS against cyberattacks.

In this paper, we propose a new methodology, based on Programmable Logic Controller (PLC) logic in order to identify cyberattacks that impacts the ICS safety. Our methodology transforms a PLC logic into a finite-state machine that represents the PLC behavior. Then, using this automaton, we identify which modifications in states of sensors and actuators leads to compromising the safety. Finally, we build attack scenarios from these events and the network vulnerabilities. We apply our methodology on a simple example, yet challenging to analyze by hand, and we show how we manage to scale up on a classical example from the control systems domain: the Tennessee Eastman chemical process.

KEYWORDS

Safety, Security, Cybersecurity, Risk assessment, Risk analysis, SCADA, ICS, IT, OT

ACM Reference Format:
Mike Da Silva, Maxime Puys, Pierre-Henri Thevenon, and Stéphane Mocanu. 2023. PLC Logic-Based Cybersecurity Risks Identification for ICS. In *The 18th International Conference on Availability, Reliability and Security (ARES 2023), August 29–September 01, 2023, Benevento, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3600160.3605067

ARES 2023, August 29–September 01, 2023, Benevento, Italy

ACM ISBN 979-8-4007-0772-8/23/08…$15.00
https://doi.org/10.1145/3600160.3605067

1 INTRODUCTION

Risk assessment is the first technical step of the IEC 27001 [10] standard on the Information Security Management Systems (ISMS) deployment. To account for new vulnerabilities or countermeasures, risk assessment has to be frequently renewed due to the state-of-the-art of attacks quick evolution. For security controls deployment, we require tools that can determine attack scenarios from risk analysis.

ICS are becoming the target of cyberattack. Often called SCADA systems, they control industrial operations like power grid management, water treatment, or transportation. Any misbehavior might possibly endanger both people and the environment due to their criticality. One of the most widely publicized attacks was Stuxnet in 2010 [18], in which a worm was able to damage Iranian nuclear centrifuges. People became aware through this incident that a computer attack can have concrete impacts on the physical world.

In the past few years, more attacks on these systems have come to light. In contrast with traditional IT systems which place a strong emphasis on secrecy and authentication of data, ICS first aim to assure availability and integrity of their process. On another hand, the lifespan of their equipment might count in tens of years, and updating them in the event of vulnerabilities is challenging. Most of protocols used by industrial systems for communication were not created with security in mind and do not provide any security feature.

The behavior of control systems is mainly managed by programmable logic controllers (PLCs). These devices allow to retrieve inputs from the physical process, update the state of the system according to a program, and compute outputs. These programs can be specified in multiple languages and using different programming paradigms. Some are textual languages such as Structured Text (ST) and Instruction List (IL), while others are graphical languages: Ladder Diagram, Function Block Diagram (FBD) and Sequential Function Chart (SFC). These languages allow to express the desired behavior of the process in the general form of state-transition diagrams.

Contributions: In this paper, we introduce a methodology to identify and generate attack scenarios against ICS. In particular, we aim to uncover scenarios where a cyberattack impacts safety functions. According to IEC 62443-1-1 [12], *security* refers respectively to the "*prevention of illegal or unwanted penetration of, or interference with the proper and intended operation of an industrial automation and control system*" and *safety* refers as "*freedom from unacceptable risk*".

In other words, safety aims to protect the system against mishaps coming from itself and security to protect the system against external threats. Thus, our goal is to determine what actions on the states of sensors and the actuators available on the industrial network an attacker can take and what effects these actions will have on the industrial process and the safety functions to ensure. We consider a number of factors, including the behavior of the process and the safety requirements that must be followed, in order to identify such attacks. However, discovering cybersecurity attacks on safety functions is a tedious process. Related works show a high tendency to run into combinatorial explosion due to the complexity of their safety and security models.

We aim our methodology to follow both an industrial risk analysis, allowing to state safety functions; and a cybersecurity risk analysis such as STRIDE [29] to find security weaknesses. To reduce combinatorial explosion, we used a lower complex model than the literature based on the PLC logic. We apply our methodology on a simple example, yet challenging to analyze by hand, and we show how we manage to scale up on a variant of a classical control system example: the Tennessee Eastman chemical process [8]. To summarize, our contributions are:

(1) A lower complex safety-security model than the literature allowing automatic analyzes scaling up to real use cases;
(2) An analysis pass to discover specific parts of the finite automaton where the actions of the attacker might compromise safety functions;
(3) A Satisfiability Modulo Theories (SMT) based analysis to find which variables should be attacked and with which values to produce attack scenarios.

The value of such scenarios is twofold. It can (i) allow to identify safety critical network connections, and (ii) identify attacks that focus on maximizing the impact on the safety functions while maintaining a legitimate PLC behavior. Such attacks would indeed hard to uncover with most of the network intrusion detection systems.

Outline: Section 2 will present and discuss related works. Section 3 will detail each step of our attack finding methodology. Section 4 will illustrate our methodology with example case studies and finally, Section 5 will conclude and show perspectives.

2 RELATED WORK

As early as 2008, Hentea [9], then Cardenas et al. [7] in 2011, identified the need to converge security and safety risk assessment methods for ICS. This topic has received an increasing interest over the years with a significant growth in proposed methods combining both safety and security of ICS. The major challenge lies in the ability to model the system under its physical (safety) and cyber (security) aspects in order to highlight the existing links between cyberattacks and their impacts on safety. Many studies try to model the system in a formal way but the complexity of the physical and cyber interactions makes it difficult to scale up.

In the related works, two trends can be distinguished to identify safety/security risks. The first one proposes a unified identification of safety and security risks based on an intersection of modeling methods. These approaches allow to provide remediation for both types of risks. Conversely, integrated methods aim to define security risks that impact safety functions for purposes of providing security remediation that protect the system's safety.

2.1 Unified Methods

Abdo et al. [1] propose to combine attack trees with a Bowtie analysis which combines fault trees with event trees. Kumar [17] describes industrial systems through a dynamic reliability block diagram augmented with attack-fault trees (AFT) to model the cascading effects of failure scenarios. This diagram is then transformed into a stochastic timed automaton to verify safety properties. André et al. [2] improves this method by proposing the possibility to weight the timed automata from the AFTs according to parameters such as budget, time or computing power of the attacker.

Kriaa et al. [16] propose a Domain Specific Language (DSL), based on the Figaro language to create a knowledge base that describes the typical components of ICS as well as their vulnerabilities and possible failures. In addition, this DSL allows the specification of rules for interactions between components alongside rules for attacks and failures occurrences. Then, after modeling the system with these components, a set of tools automatically generate attack and fault scenarios.

The benefits of these methods lies mainly in the completeness of the modeled attacks. This requirement limits these methods in two ways. Firstly, these methods depend highly on modelers's expertise, their technical sensitivities and a significant amount of technology intelligence. Secondly, the model's exhaustiveness (quality criterion) increases significantly the likelihood of combinatorial explosion. Yet, these methods generate precise attack scenarios that allow a good understanding of their consequences and cascading effects.

2.2 Integrated Methods

Puys et al. [24] introduce a method which models the system and attacker profiles through timed automata. Then, safety properties are expressed and verified by the UPPALL model checker on a composition of the system and the chosen attacker profiles' timed automata. Khaled et al. [14] improve this method by modeling the system by a composition of nodes. Each node is a component of the ICS (physical objects, software or people) defined by its capabilities and its behaviors. Then, a graph is generated by modeling the interactions and the communication protocols existing between the different nodes. Finally, from a Dolev-Yao based attacker model, an analyzer provides the possible attacks and check if they are feasible, according to a set of security properties. Cheh et al. [6] propose an optimization of [24] by considering physical layer interactions in the attacker model.

Mesli-kesraoui et al. [20] model the whole control-command chain (control program, supervision interface, physical equipment and human tasks) in a timed automaton. Then, a set of usage and safety properties are verified on the timed controller of a simple use case (a single task consisting of opening a valve) with the UPPAAL model checker.

Rocchetto and Tippenhauer [28] present a method based on the cryptographic protocol verification tool CL-Atse [30] and the ASLAN++ language. The authors extend the Dolev-Yao attaker model, notably to take into account possible physical interactions

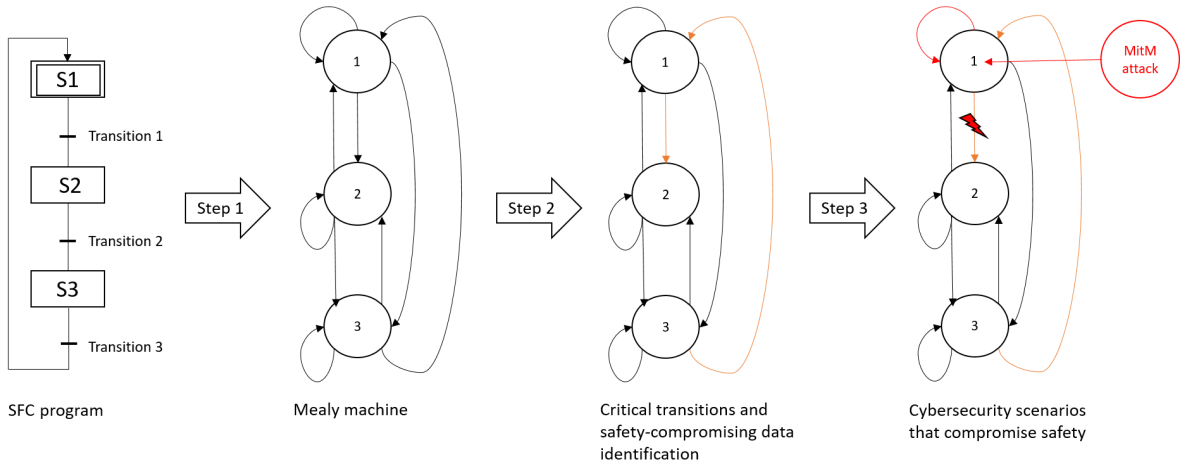


Figure 1: Method Synthesis

with the process. LTL properties on the global state of the system are specified and translated by ASLAN++ into properties that can be checked by CL-Atse. The method is applied to a use case of water treatment plant called SWaT, yet requiring a thousand lines of code and fifty entities to model.

The main limitation of current integrated methods is twofold. Formally modeling an entire system including behaviors, components, messages, etc. is time consuming and requires expertise in both cybersecurity and in formal methods to create the model properly. Secondly, even if the models are less complex than unified methods (they do not include failures), the models are still exposed to combinatorial explosion due to their high level of detail. Authors in [14] claim that they optimized their attacker model using reinforcement learning to avoid this limitation. However, they give no explanations on how this optimization is made and the size of their examples shown remain simple. For instance, modeling a physical process made of a five states automaton already generates an automaton with more than 100.000 states using their method, ineluctably succumbing to combinatorial explosion.

Related works can also be found with different goals than risk assessment. For instance, Monzer [21] designs a behavioral intrusion detection system based on hybrid automata that include a very fine modeling of the physical process and network topology. They also face a combinatorial explosion. On the other hand, McLaughlin [19] produces attack scenarios by dynamically analyzing inputs and outputs of a PLC using Buchi automata. Yet, they only consider a partial model of the physical process.

3 METHODOLOGY

We propose a safety/security risk identification method which aims to generate attack scenarios (communication interruption, packet forging, information reading, information modification that modify the behavior of the PLC logic) that compromise the infrastructure's safety. This approach is divided into three successive steps. The first one allows to convert PLC logic into a finite automaton. Then, within this automaton, we identify the data that an attacker must target in order to compromise the safety functions of the system. Finally, we determine the feasibility of these attacks by analyzing network vulnerabilities. A synthesis of the method is provided in Figure 1.

3.1 Transformation of the PLC Logic into a Finite Automaton

In this step, we aim to exhaustively explore the behavior of the PLC logic. For that, we propose to transform SFC PLC programming language, provided by IEC 61131-3, into a Mealy machine. Mealy machines are finite state transducers. Finite state means that the automaton has a finite number of state. A transducer is a specific automaton where inputs are transformed to outputs. A Mealy machine is characterized by the fact that its outputs depend on both the inputs and the current state, formally defined as follows:

- A nonempty set of inputs I_M
- A nonempty set of outputs O_M
- A nonempty set of states S_M
- An initial state $s_{InitM} \in S_M$
- Transition function δ_M: $S_M \times I_M \longrightarrow S_M$
- Output function λ_M: $S_M \times I_M \longrightarrow O_M$

Knowing that all IEC 61131-3 languages are implementation languages, it is theoretically possible to describe the behavior of the PLC in any of them. For the rest of this article, we choose the SFC language to describe our method. The SFC language allows to structure a PLC program with steps and transitions. The steps are linked together by transitions. At the beginning of the program, the initial steps are activated, i.e., the actions associated with these steps are executed. Then, the program can move from step to step through transitions which are Boolean functions to be satisfied. It is possible to write SFC transitions and actions with other IEC 61131-3 languages.

To translate SFC into a Mealy machine, we rely on the work of Provost et al. [23]. They define a process to translate Grafcet specifications into a Mealy machine for conformance test purposes. For that, they build a Stable Location Automaton (SLA) that represents all stable states of the logic described by the Grafcet and possible evolution between these states. Then, they translate this automaton into an equivalent Mealy machine. Provost et al. discuss

the difference between the Grafcet specification which describes a behavior and the SFC program which implements this behavior. They specify that the main difference lies in the evolution of the models which, in the case of the Grafcet, is instantaneous and, for the SFC, is defined by the PLC cycle time between each scan of the inputs. In our case, we do not take in account this parameter and can assume that SFC works as Grafcet.

SFC (as well as other IEC 61131-3 languages) allows to associate boolean predicates to be satisfied as conditions. Some of the boolean variables used in these predicates can be seen as the satisfaction of a condition and not as a binary variable, as could be a switch. For example, a boolean variable named X could be the result of a predicate Y > 5, where Y is a discrete value. We will show in Section 3.3 how to use a Satisfiability Modulo Theories (SMT) solver for the case of predicates involving discrete values. Now, that our Mealy machine is built and consequently the behavior of the automaton exhaustively defined, we are interested in identifying the data that can compromise the safety.

3.2 Identification of Data Compromising Safety

ICS are designed to provide a service, such as power grid management, while protecting materials, people, and the environment from harm. As mentioned above, this protection called safety, is the most important function of the system, and therefore, the one we want to secure against cyberattacks. In our method we define the safety functions to be insured in the form of logical implications between a set of inputs and outputs (e.g., $A \,\&\, \neg B \implies C$) where inputs are the antecedent and outputs are the consequent. In case of the safety function that prevent of a tank overflow, the antecedent will describe a critical state of the process (e.g tank is full) and the consequent will describe the output allowing to prevent the hazard (e.g., filling valve is closed). Then, for each of the identified safety functions, we exhaustively search for what we refer as *critical transitions* in the Mealy machine. We qualify transitions as critical when a safety protection control is executed by the PLC (modeled by the logical implication) to prevent an hazard. For the overflow example, we find when the tank is full, and thus, the filling valve command is false. We will show in this step that, upon acting on these critical transitions, it is possible for a cyberattacker to modify the behavior of the PLC and to compromise the safety of the system. The procedure to search for critical transitions is provided in the Algorithm 1.

Critical transitions define events which trigger the safety function. For example, when an electrical fault occurs (event), a circuit breaker is tripped (response) to protect the electrical network. With regard of this protection function, an attacker can disrupt the safety response. We define two types of attacks: *upstream* and *downstream* attacks. Upstream of the protection command (source state of the critical transition), the attacker can deny or alter data to mask the electrical fault, thereby compromising safety. Downstream of the protection command (destination state of the critical transition), the attacker can inject false information to force a response such as closing the circuit breaker even though the electrical fault is still occurring. Whether in upstream or downstream attacks, cyberattacks aim to distort the information coming from the process to the PLC so that it behaves in an inappropriate way.

Algorithm 1: Finding critical transitions for a safety function

```
Data: Mealy machine; Safety function
Result: Critical transitions for the safety function
begin
    M ⟵ mealyMachine;
    sIn ⟵ safetyFunctionAntecedent;
    foreach transition ∈ M do
        if sIn ∈ transition then
            AddtransitioninlistOfCriticalTransitions
        else
            next;
        End foreach
    End
```

After having identified all critical transitions for a safety function, we look for any cyberattack on data that compromise safety. In upstream attacks, we search data that allow to satisfy the critical transition, and thus, trigger the safety protection. These data are defined as AND Boolean expression into the critical transition. Thus, the attacker only needs to mask to the PLC the refreshment of one of these data to deny the safety command. This attack requires that the data we wish to compromise must change value to satisfy the critical transition (e.g. to trip the circuit breaker, the electrical fault value must change from False to True) and the emitted output is the opposite of the safety one (e.g, the circuit breaker must not be tripped). For that, we have to find every transition, pointing to our state, where the Boolean function define explicitly the value of the data that the attacker wants to compromise. These previous transitions will give us states from which the attack can be done. An example is shown in Figure 2.

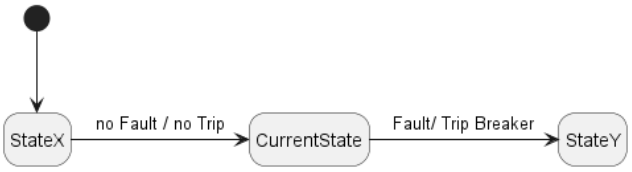


Figure 2: Upstream Attack Identification

Downstream attacks, consist in injecting data that satisfy a transition in order to force a specific output. In our case, an attacker wants to force the opposite output of safety command. For that, we look for every transitions originating from our state, where the emitted output is opposite to the safety command. Then, the attack can be realized by injecting data that satisfy one of these transitions. An example is provided in Figure 3.

3.3 Generation of Attack Scenarios that Compromise Safety

As discussed in step 2, unprotected data can lead to safety compromising cyberattacks. To ensure that these attacks are feasible, we need to identify and analyze unprotected data through a cybersecurity risk assessment. In this paper, we do not focus on classical cybersecurity risk analysis for which it exits a variety of proven

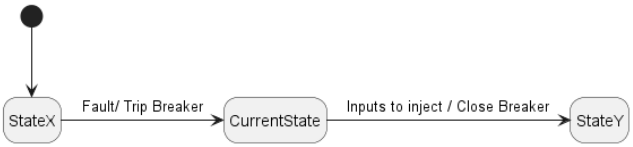


Figure 3: Downstream Attack Identification

methods such as EBIOS [3] or the IEC 62443-3-2 [13] and the IEC 27005 [11] standards.

For our method, we suppose that the attacker has access to the industrial network, where PLCs operate. For that, an attacker can, for example, connect to an internet exposed device with the help of a search engine like Shodan [1] or by spear phishing attacks like in the Ukrainian power grid attack in December 2015 [15]. Majority of the industrial protocols do not provide any security features (Confidentiality, Integrity, and Authenticity), and therefore information feedback to the PLCs and their commands can be corrupted by an attacker. In the following, we define 4 possible attacks on the industrial network:

- **Communication interruption:** In case of communication interruption, information (process data or SCADA commands) are not available to the PLC and thus cause a deny of control feedback.
- **Packets forging:** In case of packets forging, an attacker is able to set undesirable false data/commands values leading to an injection of false feedback.
- **Information reading:** Reading a data/command has no direct consequences on the PLC behavior. However, it can be used by an attacker to gain knowledge on the process and prepare attacks.
- **Information modification:** In case of data/command modification, an attacker can set process data to undesirable or false values causing an injection of false feedback.

We assume that data included in vulnerable communication channels are also vulnerable. For instance, if a communication channel does not provide authentication mechanism, a Man-in-the-Middle attack can be executed allowing an attacker to deny, tamper data and forge packets in that channel and thus deny feedback and inject false information. This will give us, all the data that the threat is able to attack and through which channel. In this paper, we focus on network attacks. Therefore, all PLC data not exposed to the network are considered as out of scope of the attacker (such as a timer between two tasks).

Then, we use these vulnerable data to perform safety compromising attacks identified in step 2. For that, we identify if the required data, for each attack, are vulnerable. If all data required by the attack are vulnerable, then the attack is considered as feasible. However, upstream and downstream attacks do not require the same vulnerabilities. Indeed, as discussed above, upstream attacks aim to mask data refreshment and downstream attacks to force an output by injecting data. Therefore, upstream attacks require a denial or a tampering of data contrary to downstream attacks that require a tampering or a forging of data. These characteristics must be taken into account during the attack feasibility checking.

Now, we can build attack scenarios that compromise safety by mapping feasible attacks with their data vulnerabilities. It is possible to build a new scenario for each vulnerability compromising the same variable. As mentioned in step 1, when building attack scenarios, we use a SMT Solver to determine if an equation containing discrete data is satisfiable and we determine, by a simplification, the range of this data. An SMT solver is a tool that determines whether a complex mathematical formula including real numbers, integers, and/or data structures is satisfiable and yield resulting values for each variable.

3.4 Evaluation results

Our methodology provides two quantitative results that help cybersecurity experts to prioritize their remediation.

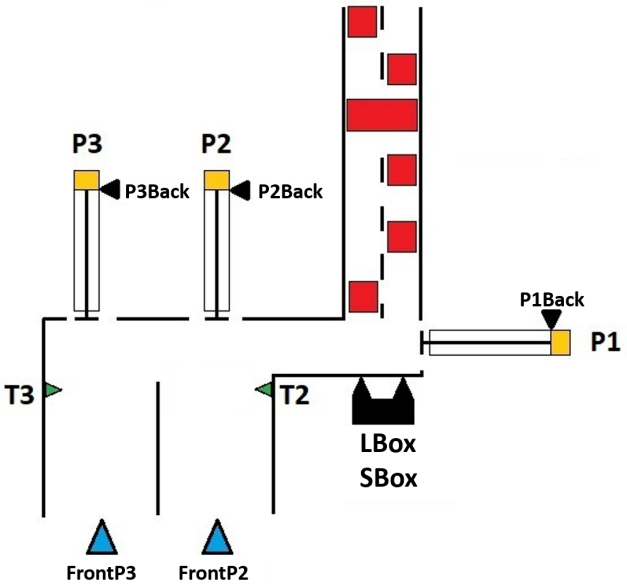
The first one is the **minimal set of data to secure** in order to protect the system against all attacks. For that purpose, we realize a Boolean minimization between all attacks scenarios to determine the minimal set of variable to secure in order to protect the system. For example, if two downstream attacks require, respectively, the variables A, C and B, C to succeed, the protection of the variable C allows to remedy both attacks. This output is appropriate for both types of attacks, nevertheless, it is more relevant for downstream attacks. Indeed, this type of attack requires that the attacker is able to simultaneously modify and/or falsify all data to satisfy the transition. Therefore, if only one of the variables is not accessible to the attacker, the attack is not feasible. Conversely, for upstream attacks, the attacker only needs to compromise a single variable to not satisfy the transition and thus for the same transition we may need to protect several, or even all, variables.

The second result is the **set of critical communication channels** (paths) of the network architecture, ranked by criticality. This criticality is determined by the variables, more precisely their frequency of occurrence and the severity of the attack scenarios that they belong, which transit inside these communication channels. Each scenario compromises a specific safety function for which we define a severity score based on its impacts. For example, a safety function that can potentially impact human lives seems more severe than a material destruction. Then, each time a data is present in a scenario, we increment a counter referring to this variable. Moreover, the value of the increment is weighted according to the scenario severity permitting to calculate the criticality of the variables. At the end, we have a ranking of the most critical variables (highest score), and thus, we determine the most critical communication channels by adding up the score of the whole variables (the weighted number of occurrences in attack scenarios) that cross it.

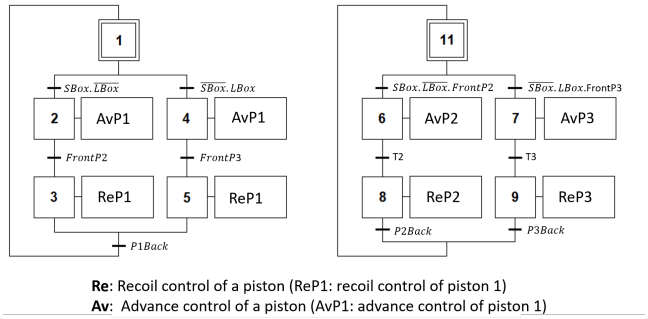
4 EVALUATION

We demonstrate our methodology on two use cases. In Section 4.1, we first introduce a simple use case to illustrate all steps of our methodology. Then in Section 4.2, we show how we scale up on a representative industrial size use case: the Tennessee Eastman chemical process [8].

[1] https://www.shodan.io/explore/category/industrial-control-systems



(a) Schematic View



(b) PLC Logic for the Use Case (SFC)

Figure 4: Selector Use Case

4.1 Box Sorting Use Case

We first describe a simple use case to depict each step of the methodology presented in Section 3. This use case, presented in Figure 4a, is a system for sorting boxes according to whether they are small or large following the logic of the PLC (SFC) presented in Figure 4b. To do this, a detector is used to determine whether the box is large or small by activating or deactivating two Boolean variables *SBox* and *LBox*. The piston P1 is then activated, pushing the box to the left. A *FrontP*2 detector will determine that the box is aligned with the T2 conveyor belt. If the box was small, the PLC logic will stop piston P1 and start piston P2. If the box was large, the piston P1 will continue to push the box towards the conveyor belt T3. When the detector *FrontP*3 is triggered, the PLC logic will stop piston P1 and activate piston P3. The pistons are controlled by 2 PLCs, each with its own logic. The first PLC controls piston 1 and the second one controls pistons 2 and 3. All the detectors and actuators are described in the Table 1.

Name	Description
P1: Piston 1	Actuator to move the boxes horizontally in front of the piston 2 and the piston 3
P2: Piston 2	Actuator to move the small boxes vertically on the conveyor belt 2
P3: Piston 3	Actuator to move the large boxes vertically on the conveyor belt 3
P1Back	Sensor to detect when the piston 1 is backwards
P2Back	Sensor to detect when the piston 2 is backwards
P3Back	Sensor to detect when the piston 3 is backwards
LBox: Large box	
SBox: Small box	Sensor to detect the size of the box (small or large)
FrontP2	Sensor to detect the presence of a box in front of the piston 2
FrontP3	Sensor to detect the presence of a box in front of the piston 3
T2	Sensor to detect the presence of a box on the belt 2
T3	Sensor to detect the presence of a box on the belt 3

Table 1: Identification Table of Detectors and Actuators

The first step of the method is to transform the PLC logic into a Mealy machine. We choose to realize a Mealy machine for each PLC, and not for the complete system (the set of the two PLCs) for the following reasons:

- The analysis of cybersecurity with respect to the overall safety of the industrial system generally leads to combinatorial explosions as described in [24] or in [14] and we consider that is more appropriate to focus on the logic of a PLC. Moreover, each PLC is generally designed to operate independently (although communicating with other PLCs), this restriction to a single PLC is consistent with the design of the industrial system logic.
- In a Mealy machine, a data is unique. However, in reality, the data will be distributed, and therefore multiplied to all the devices that use this data. In this set of possible paths of the data, some can be secured and others not, and thus the same logical data (Mealy machine) can be secured or not according to the physical context. To overcome this problem, it would be necessary to distinguish each physical context of the data, which is equivalent to distinguish each PLC in the Mealy machine.
- An actuator is driven by a single PLC. Thus, the commands (outputs) of each PLC are unique and therefore independent of other PLCs.

Our method can be extended to the global logic of the system, by performing the Cartesian product of all the PLC logics (Mealy machine) of the system. Thus, restricting to a single PLC is a performance choice to decrease the calculation and the analysis time of the results and not a limitation. However, this choice eliminates the possibility of handling distributed safety functions.

To transform SFC programs into Mealy machines, we rely on the work of [23]. As a side note, the link to the source code is obsolete but a fork of the code is available on [22]. In this particular case, each SFC step will correspond to a state in the Mealy machine

(the step 1 of the SFC will correspond to the state 1 of the Mealy machine, the step 2 of the SFC will correspond to the state 2 of the Mealy machine, etc.). For the integrity of the system, we want that the boxes are always evacuated from the conveyor belt 1 in order to avoid congestion and, a fortiori, a boxes degradation. We establish a safety function that forces the advance of piston 1 when a box is detected. We can translate this property by the implications $SBox \implies AvP1$ and $LBox \implies AvP1$. Since the method is identical for both implications, we will develop the rest of the method only with the first implication.

We are now looking for the critical transitions for the safety function we have just defined. We apply the algorithm presented in Figure 1 which consists in searching within the Mealy machine the transitions containing as input, at least, $SBox$. In our case, there are three critical transitions which link:

- State 1 to state 2 with for inputs $SBox \ \& \neg FrontP2 \ \& \neg LBox$ and for output $AvP1$
- State 4 to state 2 with for inputs $FrontP3 \ \& P1Back \ \& SBox \ \& \neg FrontP2 \ \& \neg LBox$ and for output $AvP1$
- State 5 to state 2 with for inputs$P1Back \ \& SBox \ \& \neg FrontP2 \ \& \neg LBox$ and for output $AvP1$.

Each of these critical transitions allow us to define upstream and downstream attacks. For the upstream attacks, we want the safety function not to be executed, and therefore, the critical transitions never to be satisfied. In our method, we have no knowledge of the physical process, we only know the logic of the PLC. To ensure that these attacks are not detected by NIDS, we identify the variables that are naturally in the opposite state of the critical transition because if an attacker modifies a variable himself to be in a state that does not satisfy the transition, this will probably modify the semantics of the process, and thus, will be detected by a allowlist NIDS. For example, to attack the first critical transition (state 1 to state 2) we look for all transitions that have the state 1 as the destination state, $\neg SBox \mid FrontP2 \mid LBox$ as the input value, and $\neg AvP1$ as output. Such a transition exists between states 2 and 1 which has as input $FrontP2 \ \& P1Back \ \& \neg LBox \ \& \neg SBox$ and all its outputs to False. This transition, coupled with the critical transition (state 1 to state 2), allows us to generate two attacks as follows:

(1) The industrial system evolves from the state 2 to the state 1 with the transition $FrontP2 \ \& P1Back \ \& \neg LBox \ \& \neg SBox$. An attacker blocks the refreshment of the value $SBox$. When the small box arrives, the PLC does not receive the information and compromises the safety function.
(2) The industrial system evolves from the state 2 to the state 1 with the transition $FrontP2 \ \& P1Back \ \& \neg LBox \ \& \neg SBox$. An attacker blocks the refreshment of the value $FrontP2$. When the small box arrives, the value $FrontP2$ being always at True, the PLC evolves from the state 1 to state 3 sending the order to the piston to move back and compromises the safety function.

The objective of the downstream attacks is, for our safety function, to force the stop or the recoil of the piston 1 when it starts to evacuate the box. These attacks occur from the destination state of the critical transition. For example, for the first critical transition (state 1 to state 2), the attacks will operate from the state 2. We are

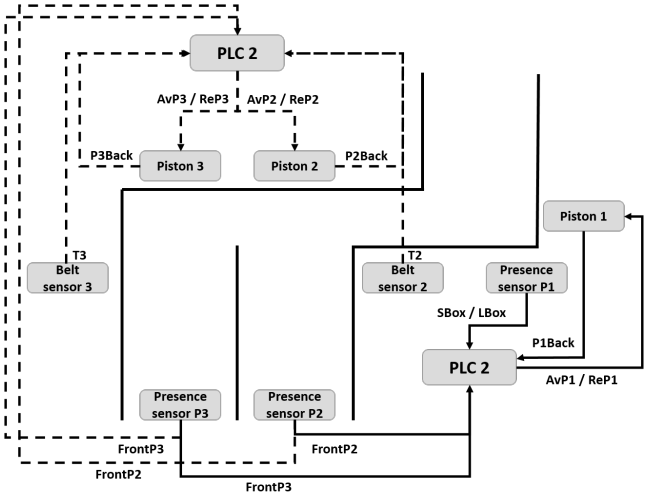


Figure 5: DFD of the Use Case

therefore looking for the set of transitions that have the state 2 as their source state and $\neg AvP1$ as output. Such a transition exists between the state 2 and 3 which has $FrontP2 \ \& \neg P1Back$ as input and $RePI$ as output. We can thus construct the following attack: The industrial system applies the safety function by evolving from the state 1 to state 2 (critical transition). An attacker injects $FrontP2 \ \& \neg P1Back$ to the PLC, forcing the transition between the state 2 and 3, which results in the sending of a backward command by the PLC, thus compromises the safety function. Knowing that this transition is legitimate, as it belongs to the Mealy machine, allowlist NIDS solutions will have great difficulty to detect these attacks. The table 2 provides the set of attacks found for the safety function $SBox \implies AvP1$ and need to be read as follows:

- **Upstream attacks**: it is possible to compromise the safety function if the header value (e.g $\neg SBox$) is maliciously maintained after one of the listed events (transitions below the header value) occurred. For instance, if the process goes from $state2$ to $state1$ (with the event $FrontP2 \& P1Back \& \neg SBox$), an attacker can maintain $\neg SBox$ value for the PLC and compromises the safety function $SBox \implies AvP1$ if the process goes from $state1$ to $state2$ (critical transition).
- **Downstream attacks**: it is possible to compromise the safety function if an attacker can maliciously force one of the listed events (transitions) after that the critical transition occurred. For example, if the process goes from $state1$ to $state2$ (critical transition), an attacker can force $FrontP2$ value to the PLC (event that force the PLC logic to go from $state2$ to $state3$) and compromises the safety function.

If we analyse the **minimal data set to secure**, obtained from our method, which permits to determine if there are data in common to all downstream attacks, we notice that all attacks require that the attacker must be able to modify or forge either FrontP2 or P1Back. In a remediation approach, cybersecurity experts will be able to prioritize their remediation by protecting the FrontP2 or P1Back data and thus protects the system from all downstream attacks.

Critical transitions	Upstream attacks	Downstream attacks
$State1 \rightarrow State2$	¬ SBox	$State2 \rightarrow State4$
	$State2 \rightarrow State1$	$State2 \rightarrow State3$
	$State4 \rightarrow State1$	$State2 \rightarrow State1(tr1)$
	$State3 \rightarrow State1$	$State2 \rightarrow State1(tr2)$
	$State5 \rightarrow State1$	
	FrontP2	
	$State2 \rightarrow State1$	
$State4 \rightarrow State2$	¬ SBox | LBox | ¬ FrontP3	same
	$State1 \rightarrow State4$	
	$State2 \rightarrow State4$	
	$State3 \rightarrow State4$	
	$State5 \rightarrow State4$	
	FrontP2	
	$State2 \rightarrow State4$	
$State5 \rightarrow State2$	¬ SBox | LBox | ¬ P1Back	same
	$State1 \rightarrow State5$	

Table 2: Attacks for the Safety Function SBox $\implies$ AvP1

We now move to the last step where we generate attack scenarios based on network vulnerabilities that we identify through a risk analysis. The data flow diagram (DFD) of the use case, depicted in 5, shows the data flows between the various devices (sensors, actuators, and PLCs) where the solid line data flows are inputs or commands of the PLC 1 and the dotted ones are inputs or commands of the PLC 2. In this diagram, the PLC 1 and the PLC 2 execute, respectively, the left and the right SFC program of the Figure 4b.

In order to correlate the vulnerable communication channels with the data that they impact, we used this DFD as follows: if, the cybersecurity risk analysis identifies that the communication protocol between the FrontP2 detector, the P1Back detector, and the PLC 1 is vulnerable to Man-in-the-Middle (MitM) attacks due to a lack of authenticity, we consider $FrontP2$ and $P1Back$ data as vulnerable to forging, reading and modification. For the following, we will consider that the cybersecurity risk analysis do not identify additional vulnerabilities, and thus, the rest of the data are secure.

With the risk analysis, we can determine what safety functions can be compromised and their impacts on the system. To do this, we filter out the set of attacks that require data only available on secured communication bus. According to the Table 2, there are only three possible upstream attacks for the data $FrontP2$ and $P1Back$ and only one for the downstream attacks (state 2 to state 3). By combining the risk analysis and the identification of critical transitions and variables, we generate attack scenarios that compromise security. For example, an attacker connected on the industrial system networks can exploit the lack of security in the communication protocol between the FrontP2 detector and the PLC to perform a MitM attack. Thus, the attacker can corrupt the $FrontP2$ value to keept it True after the transition from state 2 to state 1. When a small box arrives, the PLC will automatically order piston 1 to move backwards (from state 1 to state 3) and thus create a congestion of boxes on the conveyor belt 1.

Finally, we use the **ranked set of critical communication channels**, provided by our method in Section 3.4, if cybersecurity experts need to prioritize their remediation. In our example, we have only two unsecured data, $FrontP2$ and $P1Back$, which respectively participate in 3 (2 upstream and 1 downstream) and 2 (1 upstream and 1 downstream) scenarios. Knowing that each of these two variables have their own communication channel, we recommend to protect the communication between the FrontP2 detector and the PLC as a priority if a choice is to be made. Moreover, the **minimal set of data to secure** indicates that all downstream attacks required, at least, to corrupt $FrontP2$. Indeed, the network architecture of our use case is quite trivial, however, in a much more complex architecture where a global protection of the system is not possible (often for financial reasons) these results help cybersecurity experts to prioritize and justify their remediation by protecting the most critical communications that allow both to decrease the surface and the probability of an attack.

4.2 Results on the Tennessee Eastman Process

In this section, we aim to demonstrate that our methodology is able to scale up to representative industrial size examples. We thus perform the modeling and analysis of the Tennessee Eastman chemical process [8], which is considered as a classic benchmark for industrial system modeling and simulation. The aim of this chemical process (see Figure 6) is to synthesize two products by a chemical reaction involving four reactants. This industrial production method is based on the process of dealkylation of toluene into benzene and methane. The whole chemical reaction contains several basic reactions (mixing, separation, heating, cooling, etc.), including the production of two additional reactants. Feedback loops allow the residues to be fed back into the reaction to limit losses.

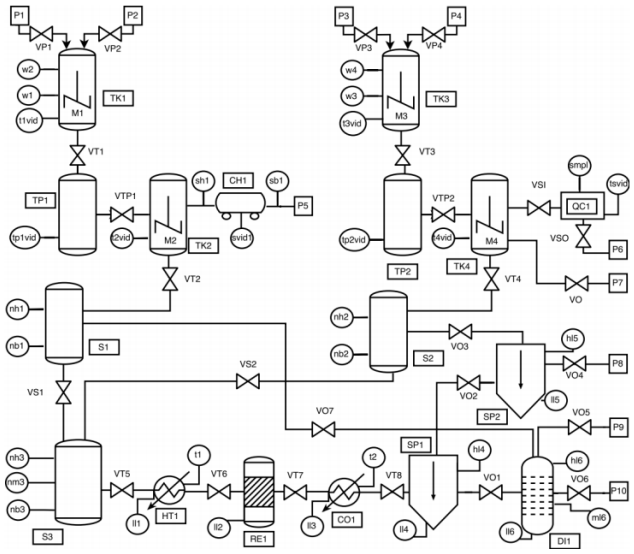


Figure 6: Representation of the Tennessee-Eastman Process

Different implementations of this process have been proposed [4, 5, 25–27]. The one we use includes close to 70 different variables, including flow rates, pressures, temperatures, levels, mole fractions,

and compressor power outputs. The operator may control over 36 of these variables (flow rates, valve positions, and reaction agitator speed) to make sure the chemical process is operating under control. Other variables are data collected from various sensors (liquid level, etc.). In our reference implementation, the process is divided into three sub-processes, each controlling a part of Figure 6. The total implementation is made of eight SFC with eight associated ladder graphs. We analyzed the model with respect to three safety functions: "preventing the overflowing of S3", "preventing the overflowing of S1", and "preventing the overflowing of S2".

Using the methodology presented in Section 3, we are able to analyze the safety of this implementation of the Tennessee Eastman process on a Dell Latitude 5500 laptop with an Intel(R) Core(TM) i5-8365U CPU @1.60GHz-1.90GHz and 16Gb of RAM in 37 seconds. Since it was implemented into three sub-processes, we analyzed them separately (in respectively 29, 7 and 1 seconds). The mealy machines associated to each of the SFCs and Ladder files contain respectively over 156, 180, and 24 states and also 15130, 15960 and 448 transitions. These states and transitions numbers only refer to the bare size of the physical process we manage to handle, prior to be crossed with the cybersecurity risk analysis. This highly contrasts with related works cite in Section 2, where most of the proposed approached only considered use cases represented with an automaton including under ten states (usually with a similar number of variables). We find 12599 attack scenarios in total for safety functions to check (respectively 7414, 5149, and 36). We analyzed only one safety function per sub-process to show quantified result but this is not a limitation according to the calculation time of attack scenarios which are respectively equal to 8, 5 and less than 1 seconds and can easily scale up to more properties. If the number of detected attacks may seem high, it is worth mentioning that we only consider a network attacker and thus protecting the network where the variables referred in the attacks will prevent them. The method results shown in Section 3.3 will help this process.

5 CONCLUSION AND PERSPECTIVES

In this paper, we proposed and applied a new methodology to identify cybersecurity risks on industrial systems. This method is based on the conversion of the PLC logic into a mathematical model called Mealy machine that allows us to exhaustively define the behavior of the PLC. Then, we showed how to exploit, from a cyberattack on the network, the legitimate behavior of this PLC to compromise safety functions. Moreover, our safety-security model, based on the PLC logic, reduces combinatorial explosion and automates the analysis. Due to the exploitation of the legitimate behavior of the PLC, the attack scenarios that we are able to identify are difficult to detect for the standard solutions used in industrial systems. Finally, our methodology provides two quantitative results: the minimal set of data to secure and the ranked set of critical communication channels for the analyzed system.

We are currently working on the development of a tool that automates this method from an XML file in PLCopen format. In the future, we plan to extend this method to automatically generate a network topology that is resistant to identified attacks while taking into account the specific constraints of industrial systems.

ACKNOWLEDGMENTS

This work was supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (IRT Nanoelec, ANR-10-AIRT-05).

REFERENCES

[1] H. Abdo, M. Kaouk, J.-M. Flaus, and F. Masse. 2018. A safety/security risk analysis approach of Industrial Control Systems: A cyber bowtie – combining new version of attack tree with bowtie analysis. *Computers & Security* 72 (Jan. 2018), 175–195. https://doi.org/10.1016/j.cose.2017.09.004

[2] Etienne Andre, Didier Lime, Mathias Ramparison, and Marielle Stoelinga. 2019. Parametric Analyses of Attack-Fault Trees. In *2019 19th International Conference on Application of Concurrency to System Design (ACSD)*. IEEE, Aachen, Germany, 33–42. https://doi.org/10.1109/ACSD.2019.00008

[3] ANSSI. 2019. *EBIOS Risk Manager*. https://www.ssi.gouv.fr/en/guide/ebios-risk-manager-the-method/ [Online; accessed 26-May-2023].

[4] Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. 2015. Revision of the Tennessee Eastman process model. *IFAC-PapersOnLine* 48, 8 (2015), 309–314.

[5] Francesca Capaci, Erik Vanhatalo, Murat Kulahci, and Bjarne Bergquist. 2019. The revised Tennessee Eastman process simulator as testbed for SPC and DoE methods. *Quality Engineering* 31, 2 (2019), 212–229.

[6] Carmen Cheh, Ahmed Fawaz, Mohammad A. Noureddine, Binbin Chen, William G. Temple, and William H. Sanders. 2018. Determining Tolerable Attack Surfaces that Preserves Safety of Cyber-Physical Systems. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, Taipei, Taiwan, 125–134. https://doi.org/10.1109/PRDC.2018.00023

[7] Alvaro A. Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. 2011. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security - ASIACCS '11*. ACM Press, Hong Kong, China, 355. https://doi.org/10.1145/1966913.1966959

[8] James J Downs and Ernest F Vogel. 1993. A plant-wide industrial process control problem. *Computers & chemical engineering* 17, 3 (1993), 245–255.

[9] Mariana Hentea. 2008. Improving Security for SCADA Control Systems. In *InSITE 2008: Informing Science + IT Education Conference*. Varna, Bulgaria, 14.

[10] IEC 27001:2023 2023. *Information security, cybersecurity and privacy protection — Information security management systems — Requirements*. International Standard. International Electrotechnical Commission, Geneva, CH.

[11] IEC 27005:2022 2022. *Information security, cybersecurity and privacy protection — Guidance on managing information security risks*. International Standard. International Electrotechnical Commission, Geneva, CH.

[12] IEC 62443-1-1:2010 2010. *Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models*. International Standard. International Electrotechnical Commission, Geneva, CH.

[13] IEC 6443-3-2:2020 2020. *Security for industrial automation and control systems – Part 3-2: Security risk assessment for system design*. International Standard. International Electrotechnical Commission, Geneva, CH.

[14] Abdelaziz Khaled, Samir Ouchani, Zahir Tari, and Khalil Drira. 2021. Assessing the Severity of Smart Attacks in Industrial Cyber-Physical Systems. *ACM Transactions on Cyber-Physical Systems* 5, 1 (Jan. 2021), 1–28. https://doi.org/10.1145/3422369

[15] Rafiullah Khan, Peter Maynard, Kieran McLaughlin, David Laverty, and Sakir Sezer. 2016. Threat Analysis of BlackEnergy Malware for Synchrophasor based Real-time Control and Monitoring in Smart Grid. https://doi.org/10.14236/ewic/ICS2016.7

[16] S. Kriaa, Y. Laarouchi, and M. Bouissou. 2015. A Model Based Approach For SCADA Safety And Security Joint Modelling: S-Cube. In *10th IET System Safety and Cyber-Security Conference 2015*. Institution of Engineering and Technology, Bristol, UK, 6 .–6 . https://doi.org/10.1049/cp.2015.0293

[17] Rajesh Kumar. 2020. A Model-Based Safety-Security Risk Analysis Framework for Interconnected Critical Infrastructures. In *Critical Infrastructure Protection XIV*, Jason Staggs and Sujeet Shenoi (Eds.). Vol. 596. Springer International Publishing, Cham, 283–306. https://doi.org/10.1007/978-3-030-62840-6_14 Series Title: IFIP Advances in Information and Communication Technology.

[18] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9, 3 (2011), 49–51.

[19] Stephen McLaughlin and Saman Zonouz. 2014. Controller-aware false data injection against programmable logic controllers. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, Venice, Italy, 848–853. https://doi.org/10.1109/SmartGridComm.2014.7007754

[20] S. Mesli-Kesraoui, A. Toguyeni, A. Bignon, F. Oquendo, D. Kesraoui, and P. Berruet. 2016. Formal and Joint Verification of Control Programs and Supervision Interfaces for Socio-technical Systems Components. *IFAC-PapersOnLine* 49, 19 (2016), 426–431. https://doi.org/10.1016/j.ifacol.2016.10.603

[21] Mohamad-Houssein Monzer. 2020. Model-based IDS design pour ICS. (Nov. 2020), 145.

[22] Ocanis. 2021. open source version of the tool Teloco. https://github.com/ocanis/Grafcet_to_Automaton Accessed: 2023-03-03.

[23] J. Provost, J.-M. Roussel, and J.-M. Faure. 2009. Test sequence construction from SFC specification*. *IFAC Proceedings Volumes* 42, 5 (June 2009), 299–304. https://doi.org/10.3182/20090610-3-IT-4004.00056

[24] Maxime Puys, Marie-Laure Potet, and Abdelaziz Khaled. 2018. Generation of Applicative Attacks Scenarios Against Industrial Systems. In *Foundations and Practice of Security*, Abdessamad Imine, José M. Fernandez, Jean-Yves Marion, Luigi Logrippo, and Joaquin Garcia-Alfaro (Eds.). Vol. 10723. Springer International Publishing, Cham, 127–143. https://doi.org/10.1007/978-3-319-75650-9_9 Series Title: Lecture Notes in Computer Science.

[25] Maxime Puys, Pierre-Henri Thevenon, and Stéphane Mocanu. 2021. Hardware-In-The-Loop Labs for SCADA Cybersecurity Awareness and Training. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*. 1–10.

[26] N Lawrence Ricker. 1996. Decentralized control of the Tennessee Eastman challenge process. *Journal of process control* 6, 4 (1996), 205–221.

[27] Cory A Rieth, Ben D Amsel, Randy Tran, and Maia B Cook. 2018. Issues and advances in anomaly detection evaluation for joint human-automated systems. In *Advances in Human Factors in Robots and Unmanned Systems: Proceedings of the AHFE 2017 International Conference on Human Factors in Robots and Unmanned Systems, July 17- 21, 2017, The Westin Bonaventure Hotel, Los Angeles, California, USA 8*. Springer, 52–63.

[28] Marco Rocchetto and Nils Ole Tippenhauer. 2017. Towards Formal Security Analysis of Industrial Control Systems. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, Abu Dhabi United Arab Emirates, 114–126. https://doi.org/10.1145/3052973.3053024

[29] Adam Shostack. 2014. *Threat Modeling: Designing for Security* (1st ed.). Wiley Publishing.

[30] Mathieu Turuani. 2006. The CL-Atse protocol analyser. In *Term Rewriting and Applications: 17th International Conference, RTA 2006 Seattle, WA, USA, August 12-14, 2006 Proceedings 17*. Springer, 277–286.